UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/712,463	11/12/2003	Judith Schwabe	P-4181CIP	9131

24209 7590 03/27/2008

GUNNISON MCKAY & HODGSON, LLP
1900 GARDEN ROAD
SUITE 220
MONTEREY, CA 93940

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
03/27/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/712,463	SCHWABE ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12/26/07*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-78* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-78* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *10/23/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 12/26/07.

As indicated in Applicant's response, no claims have been amended. Claims 1-78 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 18, 37, 56, 75 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 12, 25, 38, 51 of U.S. Patent No. 7,107,581 (hereinafter '581). Following are examples as to how the claims are conflicting.

As per instant claim 18, '581 claim 12 also recites converting first instruction to a second instruction based on relationship between base of operand, the base of the second instruction smaller than that of the first instruction, converting instructions in a *chain* such that operands with potential overflows potential beyond the base of the second base, i.e. matching by

changing type of operand of the second instruction to match with that of the first instruction base, the chain bounded by the second instruction and a third instruction being the source of the operand being subjected to the modification (see '581 claim 1); determining potential overflow associated with said second instruction and generating an output stack ('581 claim 11); indicating said 2nd instruction has potential overflow if it does not equal said first type of operand, if it does not remove such overflow, if it creates said overflow (see '581 claim 12, 1st para); indicating said 2nd instruction has potential overflow if it does not equal said first type of operand, if it does not remove such overflow, if it does not create said overflow (see '581 2nd para, claim 12: *…if overflow is not possible…*), if said 2nd instruction propagates said overflow, and if at least one operand is one input stack has potential overflow (*).

Although '581 claim 12 does not recite 'indicating said second instruction overflow if said second type does not equal said first type … indicating said second … if said second instruction does not create potential overflow … if … propagates potential overflow … at least one input stack has potential overflow' (as recited in instant claim 17), claim 18 and claim 17, together, can be addressed together with the subject matter of '581 claim 12. That is, the matter recited in instant claim 17 amounts to a slight variation of --yet equivalent to-- the subject matter of instant claim 18; hence would be treated as subsumed under claim 18 and therefore obvious when treating it with the '581 claim 12, as set forth above (see *).

'581 claim 12 does not recite '*one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input*

instruction'; but in view of '581 claim 12 reciting of 'determining potential overflow associated with said second instruction and generating an output stack based on execution of said 2nd instruction (see '581 claim 11) ... and indicating said 2nd instruction... has potential overflow, and if at least one operand ... one input stack has potential overflow (see '581 claim 12), the above limitation is considered analogously disclosed because output stack with operand being modified entails input stack operands potential overflow problem, according to the above from '581 to accomplish the above conversion operating on the operands associated with an input stack.

Nor does '581 recite '*changing the type of instructions ... to equal said 2nd type if said operand type is less than said second type*'; but '581 claim 1 recites that said second base (of a second processor) smaller than that of said first base so that the bounding of instructions between said 2nd instruction and a 3rd instruction -being a source of a potential overflow- is such that this chain involves converting to a wide base if the target operand (of the second instruction) carries a potential overflow that is beyond that of the second instruction. Hence, the above limitation is considered analogously disclosed by virtue of equivalent teaching albeit the apparent different wording in the language of the claims.

Instant claims 37, 56, and 75 recite the same limitations as claim 18 and '581 claims 25, 38, 51 recite the same subject matter of '581 claim 12; hence instant claims 37, 56, and 75 are also conflicting with the subject matter claimed in '581 claims 12, 38, 51 by virtue of the rationale as set forth above.

4. Claims 16, 35, 54, 73 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 53, 54 of U.S. Patent No. 7,107,581.

Further, **instant claims 16, 35, 54, 73** conflict with the subject matter of '581 claims 53, 54 by virtue of the input stack analysis above, i.e. output stack generating based on potential overflow of operand of 2nd instruction reads on input stack associated with 1st instruction for which a 2nd instruction of operand type belonging to lower base is to matched and converted to prevent overflow.

5. Claims 18, 37, 56, 75 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 12, 25, 38, 51 of U.S. Patent No. 7,207,037 (hereinafter '037). Following are examples as to how the claims are conflicting.

By virtue of '037 claims 12, 25, 38, 51 being substantially similar to claims 12, 25, 38, 51 of U.S. Patent No. 7,107,581 instant claims 18, 37, 56, 75 for being non-patentably distinct from those claims in '781 will be deemed obvious over '037 claims 12, 25, 38, 51 for the same reasons.

6. Claims 16, 35, 54, 73 are rejected on the ground of non-statutory obviousness-type double patenting as being unpatentable over claims 53, 54 of U.S. Patent No. 7,207,037.

By virtue of '037 claims 53, 54 being substantially similar to claims 53, 54 of U.S. Patent No. 7,107,581 instant claims 16, 35, 54, 73 would be rejected as obvious variants of '037 claims 53, 54 as set forth above.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-78 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yellin et al., USPN: 5740441 (hereinafter Yellin), and further in view of Wilkinson et al., USPN: 6,308,317 (hereinafter Wilkinson).

As per claim 1, Yellin discloses a method for arithmetic expression (e.g. col. 4, lines 42-51; *integer add* - col. 9, lines 42-52; col. 17, TABLE 1: *isub, idiv, imul, iadd*) optimization, comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction (e.g. Fig. 4B-D; col. 6, lines 6-38);

reconfigure said first instruction to operate as one operand of a second type, said second type smaller than said first type (e.g. *overflow* – Fig. 4C, 4D; *updated...modified ... by the current instructions* - col. 10, line 58 to col. 11, line 6; *and improving execution time efficiency* – col. 5, line 65 to col. 6, line 4), said reconfiguring based at least in part on the relative size of said first type and said second type (Note: overflow analysis for each successor instruction reads on size of destination place in stack for a successor operand being smaller to take required size of upper instruction in the data flow – see Fig. 4A-B; *different data types* -- col. 21, lines 16-38); and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of

instructions (e.g. Fig. 4, 5 – Note: any process that iterates one instruction at a time until all are processed reads on chain of instructions bounded by a start and a current element being processed), said chain bounded by said second instruction and a third instruction that is the source of said at least one operand (see col. 19-21: Do until there are no instructions whose changed bit is set … Do for each successor instruction …Merge the Virtual stack … Verification Success – Note: algorithm to merge – see col. 11-12).

But Yellin does not explicitly that the above reconfiguring includes *converting said first instruction to a second instruction different from the first instruction* and operable *with a one operand of a smaller type than said first type*; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type *if said operand type is less than said second type.* However, Yellin discloses adding data and rearranging stack calling structure relative to discrepancies of first and second type (see col. 1, lines 60-67; col. 20, lines 24-35; col. 21, lines 12-37; step 394, Fig. 4B) hence has suggested modifying the runtime instruction as verified by the stack to accommodate data being ported from a larger size operand --or higher platform-- to a smaller size operand -- or lower platform (see col. 5, lines 14-64; *multiple computer platforms, underlying instruction sets* - col. 1, lines 7-14). Analogous to Yellin's approach as to modifying stack runtime Java bytecodes in order to accommodate an executing platform receiving/using data coming from a platform with higher architecture base (see Yellin: Background of the invention, col. 1), Wilkinson discloses modifying stack operands from a larger base platform to operands of lesser size that would fit the target platform (see Fig. 10-11; col. 11, lines 4-48), hence disclose a form of conversion from a word operand to a byte size operand of a lesser size than the word-based

type operand. Based on the common endeavor by Yellin or Wilkinson to alleviate extraneous security and runtime resources of a given lower platform (e.g. receiving a JVM application destined for larger microcomputer into integrated circuit's microcontroller in portable devices) when loading operands for each instruction in Java method prior to runtime, it would have been obvious for one skill in the art at the time the invention was made to provide the operand type replacement as approached by Wilkinson so to support Yellin's loading process by way of converting at the receiving platform a larger type operand to a smaller size operand. One would be motivated to do so because providing operands of smaller size and familiar to the (smaller target platform) runtime as replacement for those corresponding operands of a higher size base as purported by Wilkinson and via Yellin's attempt to obviate overflow would alleviate Yellin's application resources for dealing with exception due to overflow, obviate the resources of the JVM interpreter role as desired by Yellin (see by Yellin: col. 15, lines 12-30), and according to Wilkinson, the stack space of the smaller platform (e.g. Fig. 21-24) might be loaded with data compliant to the platform thus expediting code execution via prechecking and reshuffling of bytecode, operands (see Fig. 9-11), enhancing the runtime with a safer method supporting a device whose resources are not equipped for checking large data being provided (see Wilkinson: BACKGROUND, col 3).

As per claims 2-3, see Yellin as said first instruction is arithmetic (col. 4, lines 42-51; *integer add* - col. 9, lines 42-52; col. 17, TABLE 1: *isub, idiv, imul, iadd*); a non-arithmetic, type-sensitive instruction (see col. 16-18: Table 1).

As per claims 4-5, see Yellin (in view of Wilkinson's conversion) for repeating said validating, said converting and said matching for instructions that comprise a program (see Figs 4-5) and linking each instruction to input instructions in all control paths (step 366 – Fig. 4A).

As per claim 6, see Yellin (col. 5, lines 14-64) wherein said first instruction is defined for a first processor having a first base; and said second instruction is defined for a second processor having a second base.

As per claims 7-8, Yellin wherein said first processor comprises a Java Virtual Machine (see Fig. 1-2); and in view of the obviousness rationale of claim 1 see Wilkinson (e.g. Fig. 1-2) wherein said second processor comprises a Java Card Virtual Machine with resource-restraint platform, i.e. operable with lower operand type; according to which rationale, Yellin combined with Wilkinson(see Wilkinson Fig. 11) discloses wherein said first processor comprises a 32-bit processor; and said second processor comprises a resource-constrained 16-bit processor.

As per claims 9-10, Yellin discloses said at least one input stack comprises a plurality of input stacks, said plurality of input stacks further comprising a first input stack and a second input stack; and said validating further comprises comparing operand types of corresponding entries in said first input stack and said second input stack (refer to claim 1 in light of Figs. 4); wherein said comparing further comprises indicating an error if the types of at least said first at least one stack entry and said second at least one stack entry are not equivalent (Fig. 4D, 4E, 4F).

As per claim 11, Yellin by virtue of the overflow analysis and size mismatch (refer to claim 1) and the conversion from a larger size to a smaller size in claim 1, would have rendered (based on the teaching of Wilkison) obvious the following:

setting said second type to a smallest type;

setting said second type to the type of an operand in said at least one input stack if said smallest type is less than said type of said operand (see rationale of obviousness using teaching of Wilkinson); and setting said second type to a type that is larger than said smallest type if said smallest type is greater than said type of said operand, if said operand has potential overflow, if said second instruction is sensitive to overflow and if said second type is less than said first type (see rationale of claim 1; according to which any type if predicted to overflow beyond a smaller size will be replaced, i.e. overflow analysis as by Yellin inherently teaching setting a smallest type to a larger type when said smallest type is itself greater than type of any operand potentially risking being overflowed).

As per claim 12, Yellin (in view of Wilkinson) discloses wherein said smallest type is the smallest type supported by a target processor (see rationale of claims 7-8).

As per claims 13-14, Yellin (in view of Wilkinson) discloses wherein said smallest type is the smallest type determined during a previous pass of said converting (Yellin: Figs 4-5);

wherein said third instruction is not a source instruction (any instruction being loaded does not have to be a source for a operand that is predicted to overflow); and

said changing further comprises: recursively examining input instructions until said third instruction is obtained; and setting the type of said third instruction to equal said second type (Note: in view of the recursive process to verify by Yellin and changing to a smaller operand size of the target platform by Wilkinson, any replacement to match the platform operand size being included in the verification process – see Yellin: Fig. 4-5 – will read no setting a third instruction to be equal to a second instruction previously replaced in the chain)

As per claim 15, Yellin (in view of Wilkinson) discloses (see Verification algorithm: col. 19-21) wherein said third instruction comprises a source instruction (Note: within a chain of instructions source instruction is the identified instruction starting from which replacing a original first operand with target platform operand of a lesser size begins); and said

changing further comprises: recursively examining input instructions until said third instruction is obtained; setting the type of said third instruction to equal said second type; and repeating said changing for each input instruction of said third instruction (see rationale of claim 14).

As per claim 16, Yellin discloses comprising recording, said recording comprising: determining potential overflow associated with said second instruction (see col. 9, lines 30-60); but Yellin does not disclose generating an output stack based at least in part on execution of said second instruction. But Wilkinson discloses replacement of operands for a stack (see Fig. 11, 16, 18) to support Yellin's verification endeavor as set forth in claim 1; hence the stack being modified to include (as an output stack) adjusted operands as per the combination set forth in claim 1 would also be an obvious limitation to enable Yellin to provide a smaller platform to make efficient use of its runtime environment, utilizing Wilkinson's approach for the reasons as set forth above in claim 1.

As per claim 17, Yellin discloses (combined with Wilkinson) wherein said determining further comprises: indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction

does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow (see Yellin: col. 9, lines 30 to col. 10, line 34 – Note: determining a need for change in a chain of instructions wherein operands of smaller size would be loaded in stack to *replace* – as set forth in claim 1 -- larger size operands deemed not suitable for the target platform reads on if said second instruction *does not remove* overflow or *might not necessarily create overflow*, and the automated process by which a chain in which all instructions are recursively verified reads on propagating the replacement due to potential overflow).

As per claim 18, Yellin discloses (combined with Wilkinson)wherein said determining further comprises: indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if overflow is possible based at least in part on the type of said second instruction and the relationship between said first type and said second type; and indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if overflow is not possible based at least in part on the type of said second instruction and the relationship between said first type and said second type, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow (see Yellin in view of Wilkinson – as set forth in claim 1 -- according to the rationale of claim 17).

As per claim 19, Yellin alone does not disclose creating a new output stack based at least in part on one of said at least one input stack; updating said new output stack based at least in

part on operation of said second instruction; and indicating another instruction conversion pass is required if said new stack does not equal a previous output stack. But in view of the automated and iterative verification process by Yellin combined with the operands replacement by Wilkinson as set forth in claim 1, the constant updating of stack per iteration pass in regard to what is deemed potentially conflicting in regard to stack overflow (see Yellin: Figs 4-5), it would have been obvious for one skill in the art to make many passes to update the runtime stack according to Yellin's approach using marking of unmatched operand allocation, so that combining with Wilkinson's replacement of larger size operands with more compliant size operands, Yellin's verification would be ready for execution with a stack that is repeatedly updated, and this would further alleviate runtime resources of small platforms using smaller operand type as set forth in claim 1.

As per claim 20, Yellin discloses a method for arithmetic expression optimization, comprising: step for validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

step for reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

step for matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching including a chain

being bounded; said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. Such limitation has been addressed in claim 1 using Wilkinson.

As per claims 21-38, refer to rejections set forth to claims 2-19 respectively.

As per claim 39, Yellin discloses a program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method for arithmetic expression optimization, the method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of

instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But the limitation has been addressed in claim 1 using a combination with Wilkinson.

As per claims 40-57, refer to rejections set forth to claims 2-19 respectively.

As per claims 58-76, these are the apparatus version of claims 1-19; hence are rejected with the corresponding rejections as set forth therein, respectively.

As per claim 77, Yellin discloses a method of using an application software program including arithmetic expression optimization of at least one instruction targeted to a processor, the method comprising receiving the software program on said processor, said software program optimized according to a method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand; and executing said at least one instruction on said processor;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But the limitation has been addressed in claim 1.

As per claim 78, Yellin discloses a controller configured to execute a virtual machine, the virtual machine capable of executing a software application comprising a plurality of previously optimized instructions, the instructions optimized by a method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

reconfiguring said first instruction to operate on at least one operand of a second type, said second type smaller than said first type, said converting based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above reconfiguring includes converting said first instruction to a second instruction different from the first instruction and operable with a one operand of a smaller type than said first type; nor does Yellin disclose that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But the limitation has been addressed in claim 1.

Nor does Yellin disclose that the controller is a smart card having a microcontroller embedded therein; but in view of the rationale as set forth in claim including Wilkinson's Java card machine, this smart card controller limitation would also have been obvious because of the benefits as set forth in that rationale.

Response to Arguments

9. Applicant's arguments filed 12/26/07 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 103(a) under Yellin:

(A) Applicants have submitted that Yellin purpose explicitly as by 'the present invention provides a verifier tool and method for identifying, prior to exection … would cause overflow or overflow of the operand stack … (emphasis added)" (Appl. Rmrks pg.l 21, middle) is not assertion as to change code to operate on different platform. It is highly questionable as to whether this approach by Applicants using the above emphasized statement contributes in showing how a very particular language within any specific claim has not been anticipated or rendered obvious by a corresponding cited portion in Yellin. Therefore, the above remark using Yellin's text is deemed insufficient in fulfilling a prima facie case of rebut by which a CFR 1.111(b) compliant form should be formulated (to overcome a rejection). That is, Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

(B) Applicants have submitted that Yellin describes integrity checking of downloaded programs using a bytecode verifier invoked by an interpretor prior to execution proper of said bytecodes, such that overflow is merely associated with *stack overflow* not arithmetic overflows 'as implied in the rejection' (Aprm pg. 22, middle and bottom). The rejection does not imply but has definitely pointed out where overflow can occurs in light of Yellin's real concerns with addressing complex arithmetic computation, and provided support by which arithmetic computations are involved when memory/stack overflow is to be verified.

(C) Applicants have submitted that Yellin is concerned with stopping a bytecode execution and is not 'adding data and rearranging stack calling structures relative to discrepancies of first and second types' (Appl. Rmrks pg 23, top). It is not clear whether the above remark contributes

at all in establishing how the rationale of obviousness to fulfill the 'converting said instruction to a second instruction' limitation as purported in the rejection, would have been improper. Applicants failed to show how, as proposed in the Examiner' s rationale in addressing this 'converting' limitation, this so-proposed converting step (proffered based on Yellin and the external teachings provided in Wilkinson) would be inapposite to the Yellin's endeavor or any ordinary skill level in the art when faced with the methodologies of both references. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.,* 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

(D) Applicants have submitted that the Office Action has no support for the alleged 'adding and rearranging stack calling structure ... first and second type' (Appl. Rmrks, pg. 24, middle pg. 25). The language of the claim, or any particular portion thereof, which is not even mentioned once in the above Applicant remark, should be implicated here **as basis for proving** any lack of teaching from Yellin, in order to justify how Yellin's code verifier by changing a JSR bit vector (even if it is arguably completely different for 'rearranging stack calling structure') would support Applicant's purpose to convey a Yellin's failure in fulfilling a limitation, in light of JSR bit vector or code verifier. Absent any reference to a claim particular language, the above remark has no true purpose when trying to convince that a statement (i.e. this statement being **not** a claim language) in a § 103(a) obviousness rationale has not been anticipated by Yellin's code verifier. With due respect, Applicants cannot confuse the language of the claim in regard to the cited portions of Yellin, and the language of the Examiner with such cited portions. That is,

the crux in establishing a proper rebut is about how the specific of a claimed limitation is not taught by a base reference, and in the instance of a obviousness rejection, would be for establishing how as combined the teachings suggested or taught from either or both the references used would still not fulfill the specific limitation in question. Applicants fail to provide convincing grounds as to why, for example, Yellin's modifying of the vector bit fail to substantiate for the reason for combining as set forth in the Office Action, i.e. the combined teachings to fulfill is purported to address a code conversion, not to anticipate that Yellin does teach code conversion by providing JSR bit vector, as erroneously exposed in Applicants' above remark.

(E) Applicants have submitted that for claim 1, Yellin's verifier does not convert any instruction, and by many respects unrelated to such conversion process implicating type size (Appl. Rmrks pg. 26, top). In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.,* 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

(F) Applicants have submitted that, in view of Yellin's verifying and aborting, the contribution from the second reference (i.e. Wilkinson) fails to correct such deficiency (Appl. Rmrks pg. 26, 3rd para). In response to applicant's argument that Yellin deficiency cannot be positively rectified in conjunction with a second reference, the fact that applicant has recognized, if at all, another advantage or disadvantage which would flow naturally from following the suggestion of the prior art cannot be the basis for patentability when the differences would otherwise be obvious. See *Ex parte Obiaya*, 227 USPQ 58, 60 (Bd. Pat. App. & Inter. 1985).

The differences between Yellin and Wilkinson is determined as a lack of explicit code conversion in the former, and when the context of the claim is analyzed with the same endeavor about overflow or architectural size conflict as taught or suggested in both references, a reason to combine (by one of ordinary skill) has been laid out with proper grounds to establish positive benefits or results. The argument that Yellin merely aborts cannot justify how an extended scenario using Wilkinson cannot be proposed for code reconfiguring to occur; and absent from the Applicant any valid reason as to why the references cannot be combined, or why Yellin's endeavor would clearly prohibit teaching by Wilkinson, the rejection will stand. The argument becomes mere assertion for patentability with insufficient basis to prove the 103 rationale to the contrary.

In all, the claims stand rejected as set fort in the Office Action.

Double Patenting Rejection:

(G) As per instant claim 18, The Applicants have exhibited a mapping of claim language between '581 claim 12 (Appl. Rmrks pg 2-5). The very basis of a provisional obvious-type double patenting regarding 2 inventions is that the 'conflicting' claims thereof, are analyzed based on their similar but non-identical textual form, such that obviousness can be established via a rationale based on knowledge of one of ordinary skill in the art, thereby would render the conflicting claims non-patentably distinct in order for each claimed invention to obtain a separate patent. It is deemed not sufficient to show that some particular text differs (e.g. solely mapping parallelism per textual portion of claim language) within the 2 claim sets, as proffered by the Applicants. The rationale as set forth in the Office has picked up the lack of teaching from '581 when compared against the teaching from the instant claim 18; one skill in the art when reading a

prior art context and its underlying well-known mechanism, would capture or detect a less explicit but implied methodology or teaching by inference or suggestion from the explicit reading. The office action has addressed what is missing in the instant claim 18, and has used rationale of one of ordinary skill in the art as set forth above, to put forth a rationale as to render each of the missing element (of claim 18) obvious. The Applicant fails to show how such rationale would not have been satisfactory. The recited portion in claim 12 that does not have an equivalent in instant claim 18 cannot be used as cohesively demonstrate why a feature present in '581 would not render obvious the rationale as set forth in the rejection when addressing the missing element in '581 claim; and for that matter, Applicants fail to point out why and how particularly, the feature absent in instant claim 18 (refer to claim rejection) combined with the level of one of ordinary skill in the art would NOT ALLOW '581 (including the feature not present in claim 18) to render instant claim 18 obvious over '581. Actually, the double patenting at stakes is not a USC 101 provisional rejection in which 2 sets of claims are practically identical, and comparing for exact text mapping (emphasis added) without proper evidence to show why the rationale specifically laid out in the rejection cannot be proper would amount to insufficient grounds to overcome the an obvious-type rejection.

(H) As per '581 claim 17, Applicants have submitted corresponding a&b in claim 17 cannot be treated in a gist for being same as c&d of instant claim 18; and that no teaching or suggestion for validating is alleged to be conductive to 'conversion … operands' as set forth in the rejection (Aprm pg. 6-8) in view of '581 claim 12. The endeavor as to show non-distinctly patentable invention amounts not in mapping exact parallelism in claim language but rather as detecting a common pattern of language equivalence or obvious variation thereof, and based on the context

therefrom, deriving implicit teaching that must be necessary or strongly suggested, in order to correlate what is not explicitly present or not clearly identical with the derived knowledge, and therefrom effectuate a rationale of obviousness. The rejection has addressed how the absence of c&d in '581 claim 12 can be interpreted as though c&d are obvious variation of claim 18 being subsumed in the context of a&b. The rationale as to render obvious the input stack limitation of claim 18 using the teaching of '581 claim 12 (not claim 16) would fall under the analysis as set forth above. Applicants state that piecemeal teaching from claim 16[sic] (Appl. Rmrks pg. 19 - supposed to be '581 claim 12) cannot be remote correlation that would enable imparting this stack provision into '581. The rejection has depicted proper teaching as to why this input stack would be strongly necessary; and the above remark is viewed as a mere allegation for patentability without proper reasons as to why one of ordinary skill in the art would not be motivated to see needed conception for a stack usage.

(I) Applicants have submitted textual parallel comparison, and asserted that the rejection has ignored the specifics of the claim language in lumping teaching thereof in a GIST, and is not supported with proper suggestion from the teaching at hands; and likewise, by virtue of differences shown in the mapping of '581 claim 53 versus instant claim 16 (Appl. Rmrks pg. 10-12) these differences cannot be ignored, and explanation by the Office requested. In response, the above mapping and assertion by Applicant is referred back to section G above, including the rationale as to how a portions of the limitations (c&d) can be subsumed into another (as a&b). The rejection has clearly identified which portions of the instant claim (i.e. rendering the above mapping somewhat unnecessary) have not been explicit in '581, and has provided teaching based on interpreting the context of the '581 claimed methodology in order to effectuate obviousness

reasoning. It is deemed superfluous to have to recopy the textual content of the rejection back in the current section of Examiner's response. Further, since a Java smart card is but a mere smaller subset of the broader bytecode execution context of the instant claim, the argument about Java Smart card is non persuasive (Appl. Rmrks pg. 12). The arguments that input stack has never sufficient inferred in '581 claim 53 will be referred to section H above.

(J) Applicants have provided mapping of '037 claim 12 versus instant claim 18 (Appl. Rmrks pg. 14-17). The rejection is not founded upon mapping of mere textual limitations as would a proper USC 101 provisional rejection. The arguments is referred back to section G.

(K) Applicants have provided mapping of '037 claim 53 versus instant claim 16 (Appl. Rmrks pg. 18-19) using the same argument to assert distinction between the claimed invention. As clearly provided in the rejection, each limitation not explicit in '037 or '581, the rejection has provided how one of ordinary skill in the art would construe a underlying context or teaching by which additional feature deemed necessary are **derived**; that is, the obviousness has been based on level of ordinary skill in the art that flows along with the full or partial teaching of the prior art, including inherent, highly necessary, or implied features purported for a similar problem solving or domain of endeavor. For example, Java bytecodes along with its closely association with stack-based execution in JVM was a well-known concept, the endeavor of which is clearly suggested or partially depicted in both '581 and '037; such that based on the memory issues or problem solving for a Java runtime support in terms of overflow or instruction size mismatch or memory allocation adverse effects as perceived from respective claimed invention, the rationale to correlate overflow and use of stack has been derived (refer to rejection). With due respect, it

would be deemed superfluous to repeat the whole text of the respective rejection back in this section.

The Double Patenting rejection will stand as set forth in the Office Action.

Conclusion

10. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before

using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

March 20, 2008